ÖREBRO UNIVERSITET

INSTITUTIONEN FÖR
NATURVETENSKAP OCH TEKNIK

Datorteknik, DT111G

Datorteknik, teori

A001
4,5 högskolepoäng

Skriftlig tentamen

2022-08-24

Programmering grundkurs, DT111G (A001)

Tillåtna hjälpmedel: penna, radergummi, engelska-svenska ordbok

Instruktioner:

- ***Läs igenom alla frågor noga.***
- **Ange tentamenskoden på svarsdokumentet.**
- **Du kan svara på *Svenska* eller *Engelska*.**
- ***Skriv tydligt* (gäller även för en digital tentamen).**
- **Detta är en individuell examination - alla misstankar om otillåtet samarbete kommer att rapporteras.**
- **Ansvarig lärare finns tillgänglig via telefon fr.o.m. andra skrivtimmen.**
- **Skriv läsligt!**
- **förklara och motivera era svar**

Ansvarig lärare: Pascal Rebreyend, tel: 0702001422

För betyg G krävs 50% av total poäng (20 på 40)

(26 poång gav betyg 4, och 32 poäng gav betyg 5)

Lycka till!

Question 1 (2 points)

You wrote the following piece of code:

```
struct mystruct {
      int pos;
      char tag1;
      double x;
      char tag 2;
      double y;
};
```

but the compiler decides to rewrite your code into:

```
struct mystruct{
      double x;
      double y;
      int pos;
      char tag1;
      char tag2;
};
```

Why the compiler rewrote your code?

Question 2 (4 points)

You are writing some C-code to do some matrix (bi-dimensional) computations. What are the different possibilities you may have by using a malloc or calloc to store the values of the matrix in memory. Explain why and how you may carefully choose between them.

Question 3 (3 points)

Caches can speed up the execution of the code by caching in their small but fast memory data which are stored on slower memory like RAM or Hard-drives when the same data is read and written several times during the execution. But why on some system the cache is also able to speed up the process when data are only read once?

Question 4 (4 points)

Here some assembly code for an imaginary ARM CPU.

```
      load R3,#0
LAB1: cmp R3,R6
      bne LAB2
      jsr X
      add R3,#1
      br LAB1
LAB2: load R3,R6
      mul R3,R3,R6
```

This assembly has the following instructions:
load x,y: load the value of y into x.
cmp a,b: compare a and b and update the test register
bne x: Branch to x if not equal
jsr x: jump to subroutine x
mul a,b,c: put in register a the result of a*b
Br x: branch (goto) x
The # sign indicates an immediate value.

What is done by the assembly code above? Can you do some reverse engineering and have ideas about the corresponding C code?

Question 5 (2 points)

On a classical computer like a PC, is it possible to write a full software by using different programming languages for the different parts (C, Fortan, assembly) and how you will do it in such case?

Question 6 (3 points)

- How the decimal number 120 is coded as a 32 bits number, big endian, using the 2's complement representation?
- Same question with -32 instead of 120
- How 105 is coded using 1's complement?

Question 7 (4 points)

We have a computer with a Direct Mapped Memory Cache. The main memory is 16GB and the cache is 64KB. The size of a block is 512B. Explain what a Direct Mapped Memory Cache is and how it works in this particular example.

Question 8 (4 points)

You have the following C-code:

```
.....
char *ptrc;
double *prti;
.....
ptrc++;     //line A
prti++;      //line B
.......
```

Lines A and B increment two variables. But if you analyze what is done by the CPU when executing these 2 lines, which differences you will discover?

Question 9 (4 points)

When translating a C code or similar into assembly, what are the different techniques used or which can be used to pass arguments and returns values. Describe pro and cons of each techniques.

Question 10 (2 points)

What is a microcode and what its usage is?

Question 11 (3 points)

We use the term CPU for a wide range of microprocessors: From the latest and powerfull cpu used in your new desktop or laptop up to a small and cheap chip used in very small embedded devices performing very simple tasks. If you want to run a operating system such as windows or linux, aside the power, which features are needed on a cpu and why?

Question 12 (2 points)

Is it possible to design and built a processor for which in the assembly all instructions have at the max 1 operand? Same question with 0 operand? If yes, how we can do operations like additions or multiplications?

Question 13 (3 points)

What is a pipeline in a processor and how it works? Why it is an important feature and what should we take care of as user in order to get the most benefit of it?